

# 本周周报（4.29-5.5）

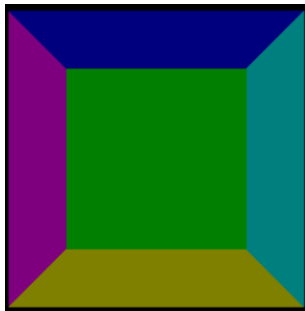
刘昊南

## 本周工作

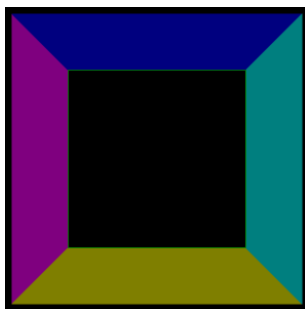
- 在对上周实现的 `depth peeling` 进行的进一步测试过程中，发现了一些 bug
  - 上周的简单测试程序使用了几个与视线方向垂直的面来进行测试，结果是正确的。现在使用一个立方体来进行测试，出现了一些问题，即对一个立方体剥离了很多层仍没有剥离完  
立方体正面与视线方向垂直，第一层的结果是立方体的正面，是正确的



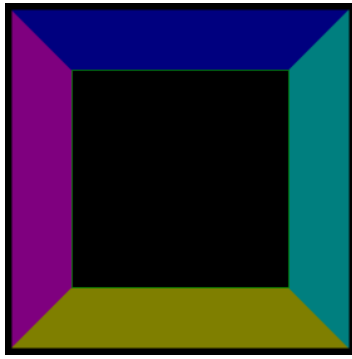
第二层的结果是立方体的背面和上下左右四个面，也是正确的



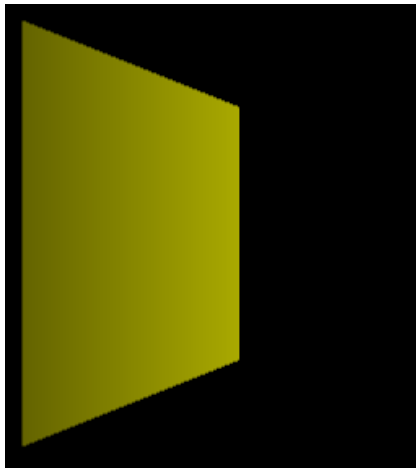
剥离本应在第二层结束，第三层的结果应该是背景，但是实际的结果却是上下左右四个面



一直持续到第十层仍未剥离干净



说明在对不与视线方向垂直的面的剥离过程中出现了问题，任画一个不垂直于视线的平面，会出现上述问题，下面是剥离十层的结果，矩形颜色的红色分量为当前 pass 的片元深度值，绿色分量为存储在上一个 pass 的深度纹理中的深度值，蓝色分量为 0



2. 经过仔细研究，发现上图的矩形颜色是由暗到亮标准黄色，说明当前 pass 的片元深度值与保存在上一个 pass 的深度缓冲里的深度值应该十分接近，但是又不完全相等，存在一个误差使得保存在深度缓冲里的深度值略小。在 depth peeling 的 fragment shader 中将抛弃片断的代码做了如下修改后，这个 bug 便消失了

```
float depth = texture2D(depthTex, pos).x;  
if(gl_FragCoord.z <= depth+0.001) //修改  
    discard;  
gl_FragColor = vec4(gl_FragCoord.z, depth, 0.0, 1.0);
```

3. 查阅了一下 OpenGL 的有关资料，发现深度缓冲区的精度只有 24 位，而 fragment shader 中的片元深度是 32 位的浮点数，由于深度保存到深度缓冲区是被截断了，只能精确到小数点后 3 位，导致缓冲区里的深度比真实的小，所以出现了上面的 bug。在上周的测试中，使用的垂直于视线的平面的深度值没有精确到小数点后 3 位，所以没有发现这个问题。而在绘制不垂直于视线的平面时，插值出的深度值用到了小数点后第六位，所以反应出了这个 bug。

## 下周计划

1. 使用 32f 的单通道颜色纹理来替代深度纹理使用，相信应该可以彻底解决这个 bug

2. 与海东师兄讨论重构 `scenemanager` 类的方案, 对 `scenemanager` 类进行重构, 加入 `depth peeling` 功能进行混合绘制